Analysis of College Completion at Institutions in the United States

GOUTHAM CHADALLA MANJUNATH, UBIT-gouthamc UB-Number-50466616
MOHIT MAHESH VASWANI, UBIT-mohitmah UB-Number-50468309
SHREYAS ATHREYA VENKATESH, UBIT-venkate7 UB-Number-50366325

ACM Reference Format:
Goutham Chadalla Manjunath, Mohit Mahesh Vaswani, and Shreyas Athreya Venkatesh. 2022. Analysis of College Completion at Institutions in the United States. 1, 1 (December 2022), 20 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROBLEM STATEMENT

The data-set that we have chosen for this project is College Completion dataset. The purpose of selecting this dataset is that students would make informed decision before choosing a degree program based on the degree completion rates in various universities. Additionally, Universities could use this dataset to improve their degree programs, understand the demographic of the students. Universities could also make changes to their degree offering based on the student completion rates of specific degrees. Furthermore, the federal and State education departments can take note of the demographic data and make appropriate resource allocation to uplift those Universities with greater dropout rates.
A database system supports multiple tables whereas excel files have limited support for the number of tables. A database supports the integrity of data whereas in excel files it is limited. Storing data in a database, therefore, promotes data being stored centrally which in turn leads to data consistency

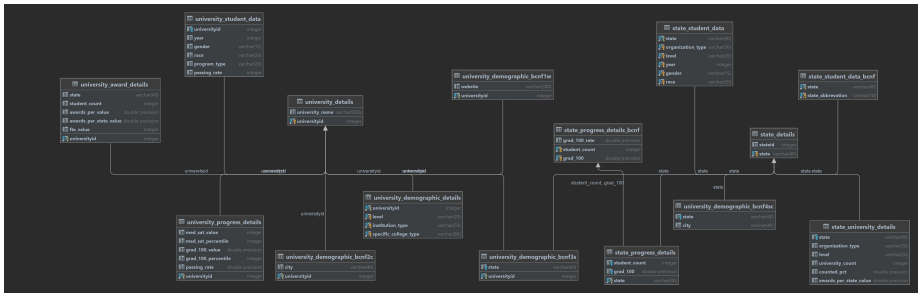
2 TARGET USER

- The User - The Main targeted user for this information are students that would be embarking on their journey for a degree program. This data would help such students make an informed decision on what degrees to choose from, and which university to settle on based on the graduation rates.
- The administers - The administer will be the universities/University board/ websites like US News. Additionally, the Education department of the Federal and State government could use such a database for better resource allocation.
- Real-life scenario - Situations where students wish to join the universities which provide a maximum probability of success rate combined with location preference.

Authors' addresses: Goutham Chadalla Manjunath, UBIT-gouthamc UB-Number-50466616; Mohit Mahesh Vaswani, UBIT-mohitmah UB-Number-50468309; Shreyas Athreya Venkatesh, UBIT-venkate7 UB-Number-50366325.

Manuscript submitted to ACM

3 ER DIAGRAM

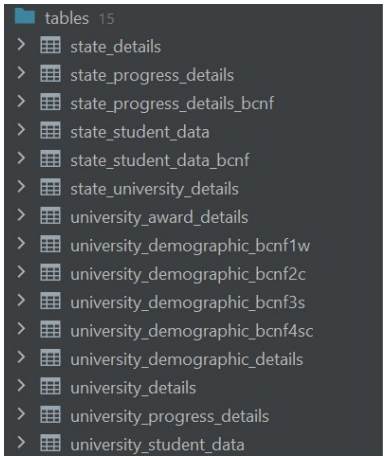


4 DATABASE IMPLEMENTATION

4.1 Data Schemas

In this section we will be presenting the schemas that are present in our dataset. In all this dataset includes nine tables or nine relational schemas.

The Following is the image of the tables from our pgadmin for our Dataset.



From the ER diagram we can see that the information about students that are currently enrolled in degree programs at Universities.

This table provides the student gender race distribution among the student by including attributes like gender and race. Most importantly, it includes the passing rate of the students, which serves as the primary quantifying metric for this dataset.

The above ER diagram includes information about the university's demographic data. This table provides the geographical data such as the state and the city in which a particular university is located. It also provides the type of institution which could be Public or Private. Under private it could be for profit or not for profit. The dataset also provides the website of such institutions for easy access to the details. The relation also has university id which servers as the unique identifier.

The above relation provides the information about the statewise information of the progress details for the institution information which has the type of program. Also it has grad_100 which shows the graduation as expected time. and grad_100_rate which is similar to the previous information but provides the rate as well.

The relations shows how any state fairs as per the completion rate of student in their respective degree programs and levels and organization type. This relation provides a quantification of the state information.A detailed description of each attribute is give in the attributes section.

This relation is used just to connect relations which either have stateid or state information.

The relation like the previous relation which showed the state distribution of students and their completion rates, this provides the similar information with a hint of student demographic information.

This relation is again a linking table with provides the university name with the appropriate university id which servers as a lookup table.

The university awards table provides the statewise completion rate of students who have qualified their course in one attempt. The awards per value and awards per state value both provide the completion rate as per the state and the national average.The fte value provides the total full time graduates followed by a unique identifier that is the universityid.

The University progress details provide very different information seen so far. This table provides the information about information about sat scores, sat percentiles the graduation rate followed by the graduation percentile rate with a unique identifier as the universityid.

Relationships between tables - The **unversity_details** the table contains university_name and universityid. The **university_awards_details** will contain all the information about awards for a particular university. **university_student_data** will contain demographic information about student data of that university. **university_progress_details** contains all the information about sat exam score eligibility, passing rate, etc for a particular university.**university_demographic_details** will contain details about the institution type, website, state, etc for a particular university. **state_details** contains stateid and statename for a particular state. **state_progress_details** contains data about program types offered, graduation rate, etc for a particular state. **state_university_details** will store different types of organizations, university count, and awards earned by universities in that state.**state_student_data** contains demographic details of student studying in that state.

4.2 Primary and Foreign Keys

- The relation **university_details** has universityid is the primary key.
- The relation **state_details** has stateid is the primary key.
- The relation **University_student_data** has the universityid that serves as primary key. It is also the foreign key which is referred from relation **university**$_detail$.
- The relation **university_demographic_details** has the universityid that serves as the primary key. The foreign key of this relations is state and univesityid. state is refereed from **state_details** and univesityid is referred from **univesity_details**.
- The relation **state_student_data** has the state that serves as the foreign key which is referred from **state_details**.

- The relation **university_progress_details**has the universityid that serves as primary key. It is also the foreign key which is referred from relation **university**$_detail$.
- The relation **university_award_details** has the universityid that serves as primary key. It is also the foreign key which is referred from relation **university**$_detail$.
- The relation **state_progress_details** has the state that serves as the foreign key which is referred from **state_details**.
- The relation **state_university_details** has the state that serves as the foreign key which is referred from **state_details**.

4.3 Data Source

The data for this project was collected from the following link below,
Data.world. 2022. College Completion - dataset by databeats. [online] Available at: <https://data.world/databeats/college-completion> [Accessed 16 October 2022] Manuscript submitted to ACM.

4.4 Deviation from Milestone-1

We have the following modification on the relations from checkpoint 1.

- **state_student_data** relation had the following attribute set{state, state_abbreviation, organization_type, level, year, gender, race}. We have the following functional dependency in this relation which causes the relation to not be in BC-NF form,
  state $\rightarrow state\_abbreviation$
  $state\_abbreviation \rightarrow state$

So, we decompose the relation into BCNF with the following candidate keys, state, organization_type, level, year, gender, race state_abbreviation, organization_type, level, year, gender, race
Now we form two tables from state_student_data relation,
state_student_data and **state_student_data_bcnf**

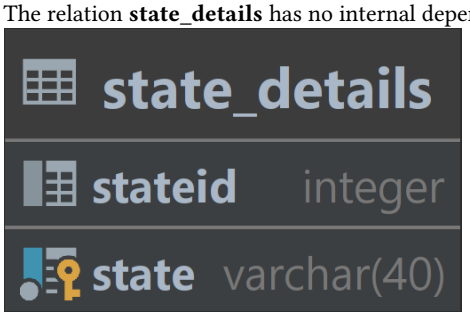
- **state_progress_details** relation had the following attribute set {state, state_count, grad 100, grad_100_rate}. We have the following functional dependency in this relation which causes the relation to not be in BC-NF form,
  state $\rightarrow state\_count, grad100$
  $state\_count, grad100 \rightarrow grad\_100\_rate$
  Now we also observe a transitive dependency, state can determine state_count grad 100, and state_count and grad 100 can determine grad_100_rate which causes transitive dependency.
  Therefore, we decompose the relation into the following relations, **state_progress_details** and **state_progress_details_bcnf**
- **state_university_details** relation had the following attribute set {state, state_abbreviation, ,organization_type, level, university_count, counted_pct, awards_per_state_value}. We have the following functional dependency in this relation which causes the relation to not be in BC-NF form,
  state $\rightarrow state\_abbreviation$
  $state \rightarrow unicount$
  So, we decompose the relation into BCNF with the following candidate keys,
  {state,organization_type, level, university_count, counted_pct, awards_per_state_value},

{state_abbreviation,organization_type, level, university_count, counted_pct, awards_per_state_value}
Now we form two tables from state_university_details relation,
state_university_details and **state_university_details_bcnf**

- **university_demographic_details** relation had the following attribute set {city, state, level, institute_type, specific_college_type, website, universityid} We have the following functional dependency in this relation which causes the relation to not be in BC-NF form,
universityid → *website*
universityid → *city*
universityid → *state*
So, we decompose the relation into the following BC-NF reduced tables.
university_demographic_details
university_demographic_bcnf1w
univerity_demographic_bcnf2c
university_demographic_bcnf3s
univerity_demographic_bcnf4sc

4.5 BCNF

- The relation **state_details** has no internal dependency and hence it is BCNF.

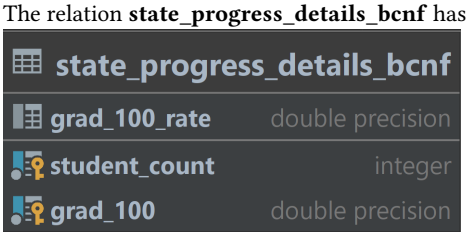


The state is a candidate key that determines every other attribute.
The relations state_details do not have multi-valued attributes hence it is in 1NF.
The relations state_details do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency, and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **state_progress_details** has no internal dependency and hence it is BCNF.

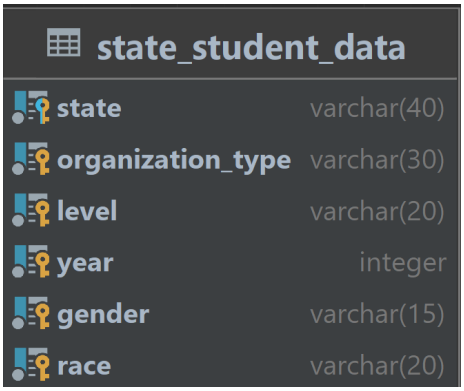
State is a candidate key that determines every other attribute.
The relations state_progress_details do not have multi-valued attributes hence it is in 1NF.
The relations state_progress_details do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency, and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **state_progress_details_bcnf** has no internal dependency, and hence it is BC-NF.



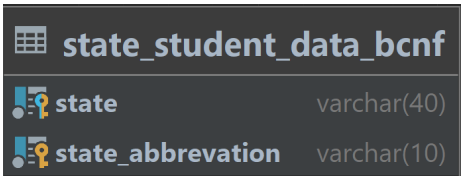
grad_100 and student_count are candidate key that determines every other attribute.
The relations state_progress_details_bcnf do not have multi-valued attributes hence it is in 1NF.
The relations state_progress_details_bcnf do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency, and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **state_student_data** has no internal dependency and hence it is BC-NF.



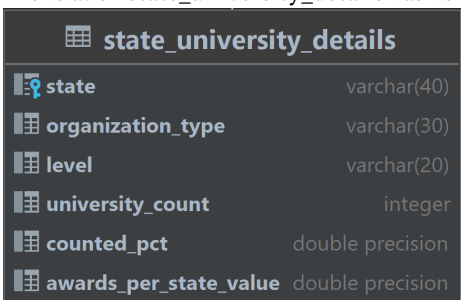
state, organization_type, level year, gender, race are candidate key.
The relations state_student_data do not have multi-valued attributes hence it is in 1NF.
The relations state_student_data do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency, and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **state_student_data_bcnf** has no internal dependency and hence it is BCNF.



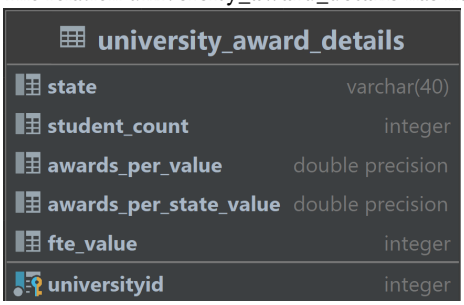
state is the candidate key that determines every other state_abbrevation.
The relation state_student_data_bcnf do not have multi-valued attributes hence it is in 1NF.
The relation state_student_data_bcnf do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **state_university_details** has no internal dependency and hence it is BCNF.



state is a candidate key that determines every other attribute.
The relations **state_university_details** do not have multi-valued attributes hence it is in 1NF.
The relations state_university_details do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency, and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

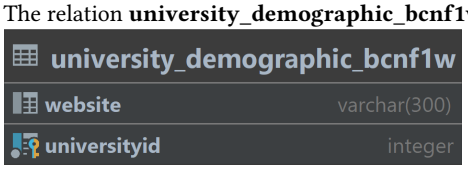
- The relation **university_award_details** has no internal dependency and hence it is BCNF.



universityid is the candidate key that determines every other attribute.
The relations university_award_details do not have multi-valued attributes hence it is in 1NF.
The relations university_award_details do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.

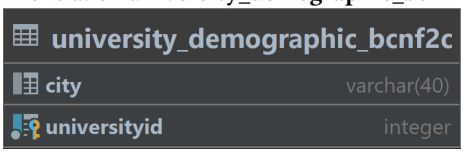
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_demographic_bcnf1w** has no internal dependency and hence it is BCNF.



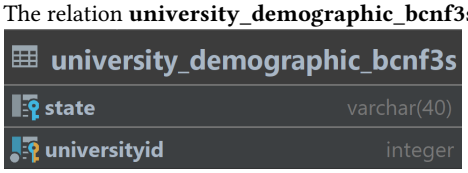
universityid is the candidate key that determines website.
The relation university_demographic_bcnf1w do not have multivalued attributes hence it is in 1NF.
The relations university_demographic_bcnf1w do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_demographic_bcnf2c** has no internal dependency and hence it is BCNF.



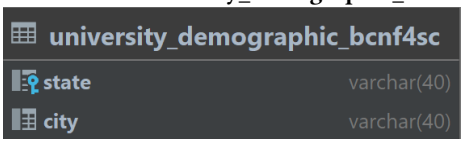
universityid is the candidate key that determines city.
The relations university_demographic_bcnf2c do not have multi-valued attributes hence it is in 1NF.
The relations university_demographic_bcnf2c do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_demographic_bcnf3s** has no internal dependency and hence it is BCNF.



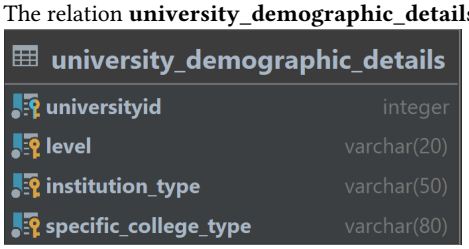
universityid is the candidate key that determines state.
The relations university_demographic_bcnf3s do not have multivalued attributes hence it is in 1NF.
The relations university_demographic_bcnf3s do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_demographic_bcnf4sc** has no internal dependency and hence it is BCNF.



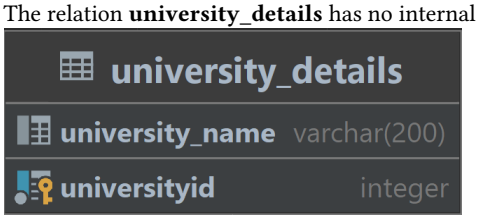
State here is the candidate key which can determine the city of university.
The relations university_demographic_bcnf4sc do not have multivalued attributes hence it is in 1NF.
The relations university_demographic_bcnf4sc do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_demographic_details** has no internal dependency and hence it is BCNF.



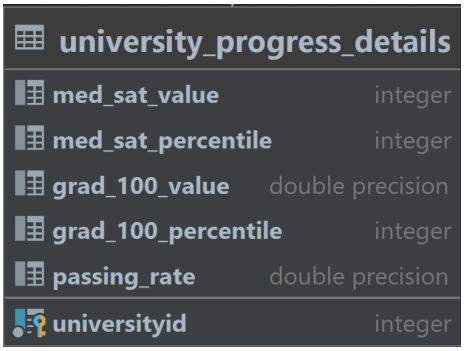
Universityid,level,institution_type,specific_college_type together here is the candidate key which can determined.
The relations university_demographic_details do not have multivalued attributes hence it is in 1NF.
The relations university_demographic_details do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_details** has no internal dependency and hence it is BCNF.//



Universityid here is the candidate key which can determine the university_name.
The relations university_details do not have multivalued attributes hence it is in 1NF.
The relations university_details do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

- The relation **university_progress_details** has no internal dependency and hence it is BCNF.



Universityid here is the candidate key which can determine the other attributes of table.
The relations university_progress_details do not have multivalued attributes hence it is in 1NF.
The relations university_progress_details do not have partial dependency hence it is in 2NF.
Additionally, we do not have a transitive dependency and hence it is in 3NF.
Since we do not have mixed partial dependency, hence it is also in BC-NF.

4.6 Attributes Information

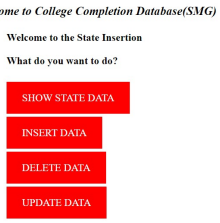
- From the ER diagram we can see that from the relation **university_student_data**. The attributes included in this schema are the **universityid** that serves as the unique identifier for students in this database. The universityid has datatype as *integer*. The attribute **year** shows the enrollment year for a particular student which is of an *integer* datatype. The attributes **gender** and **race** are both *varchar* with sizes 15 and 30 respectively. The attribute **student_count** involves the number of students which can be one of the following bachelor's degrees, associate degrees, and certificate programs and has a datatype *varchar* with size 20. The attribute **passing_rate** shows the primary metric of this dataset which is the particular students passing rate which has the datatype as *integer*.
- The relation **university_demographic_details** provides the following attributes. The attribute **city** and **state** provides the location details has the datatype *varchar* 40. the attribute **level** has the datatype *varchar* 20. **institution_type** and **specific_college_type** are both *varchar* with sizes 50 and 80. **website** gives the university website which has the datatype as *varchar* with size 300. Finally **universityid** uniquely identifies the the type of university which has the datatype *integer*.
- **state_progress_details** has the attributes **state** which has the datatype as *varchar* of size 40 followed by the **program_type** which is also of datatype *varchar* with size 30. The attributes **grad_100** and **grad_100_rate** both are *float8* datatype which provides the information about how the student pass their program ideally and their rates.
- The relation **state_university_details** has the attributes **state** which has the datatype *varchar* with size 40 followed by its shorthand **state_abbrevation** of type *varchar* of size 10. **organization_type** which specifies if the university is public or private. If it is private either it is for profit or not for profit. The data type for this is *varchar* with size 30. Now the attribute **level** specifies if it's a 4-year degree program or a 2-year degree program which has datatype *varchar* of size 20. **university_count** specifies how many of such universities are present in a particular stateid which has the datatype *integer*. The attribute **counted_pct** has the information

about how many first time student have entered the degree program which has the datatype *float8*. The **awards_per_state_value** has the datatype *float8* shows how many of the students who started the degree have complete the degree at their first attempt.

- **state_details** is only used to connect tables which have only stateid and tables which have states.
- The relation **state_student_data** has the attributes state which has the datatype *varchar* which serves as an identifier of this relation. Furthermore, we have the shorthand **state_abbreviation** for a state which was also there in a previous relation with datatype *varchar*. The **level** provides the level of education which can be a 4-year or 2-year program. Additionally, we have the **organization_type** which specifies if the if it is a public, private for profit, or not as mentioned in earlier relations as well. The attributes **gender** and **race** of type *varchar* provide the information state-wide followed by the year of induction for the cohort/student.
- The next relation is the **university_details** servers as a lookup for university names and universities its similar to the state details.
- The relation **university_award_details** is the distribution of the completion rate with the attributes **awards_per_value** of type *float8* **awards_per_state_value** of type *float8* and the **fte_value** of type *integer* which serves as the indicators for completion success-rate with respect to state, national and total completion rates. Also, the **universityid** is the unique identifier with type *integer* as mentioned earlier.
- **university_progress_details** is very different from the previous attributes with has a quantifier to the completion rates based on the SAT scores **med_sat_value** which is of type *integer* followed by **med_sat_percentile** which is of type *integer* which provides percentile quantification, which is followed by the **grad_100_value** and **grad_100_percentile** of types *float8* and **passing_rate** of type *integer*, all three of them serve the purpose of defining the progress of a university on the basis of the completion rate of student undertaking the degree program with the **universityid** which links this to the respective universities. All other tables are BC-NF forms created by decomposing the core eight tables defined in checkpoint-1 submission.

5 USER INTERFACE-UI

The following is our welcome page,



We initially have the following data in our user interface,

Nevada	32
New Hampshire	33
New Jersey	34
New Mexico	35
New York	36
North Carolina	37
North Dakota	38
Ohio	39
Oklahoma	40
Oregon	41
Pennsylvania	42
Rhode Island	44
South Carolina	45
South Dakota	46
Tennessee	47
Texas	48
Utah	49
Vermont	50
Virginia	51
Washington	53
West Virginia	54
Wisconsin	55
Wyoming	56
Gujarat	88
Punjab	188

Back To HOME!!!

Now we show the CRUD operations,

We create an entry for **Kerala** with state-ID as **500**.

Here is the UI operation,

State Name: Kerela

State ID: 500

Insert

SHOW STATE

BACK TO HOME!!!

The result of the data after the operation

New Hampshire	33
New Jersey	34
New Mexico	35
New York	36
North Carolina	37
North Dakota	38
Ohio	39
Oklahoma	40
Oregon	41
Pennsylvania	42
Rhode Island	44
South Carolina	45
South Dakota	46
Tennessee	47
Texas	48
Utah	49
Vermont	50
Virginia	51
Washington	53
West Virginia	54
Wisconsin	55
Wyoming	56
Gujarat	88
Punjab	188
Kerela	500

Back To HOME!!!

We Update the record **Kerala** to have state-ID as **999**.

Here is the UI operation,

Insert State Details to UPDATE

State Name: Kerela

State ID: 999

Update

SHOW STATE

BACK TO HOME!!!

The result of the data after the operation

Nevada	32
New Hampshire	33
New Jersey	34
New Mexico	35
New York	36
North Carolina	37
North Dakota	38
Ohio	39
Oklahoma	40
Oregon	41
Pennsylvania	42
Rhode Island	44
South Carolina	45
South Dakota	46
Tennessee	47
Texas	48
Utah	49
Vermont	50
Virginia	51
Washington	53
West Virginia	54
Wisconsin	55
Wyoming	56
Gujarat	88
Punjab	188
Kerela	999

Back To HOME!!!

Here is the UI operation,

Insert State Details to Delete

State Name: Punjab

State ID: 188

Delete

SHOW STATE

BACK TO HOME!!!

We Delete an entry for **Punjab** with state-ID as **188**.

The result of the data after the operation.

Nebraska	31
Nevada	32
New Hampshire	33
New Jersey	34
New Mexico	35
New York	36
North Carolina	37
North Dakota	38
Ohio	39
Oklahoma	40
Oregon	41
Pennsylvania	42
Rhode Island	44
South Carolina	45
South Dakota	46
Tennessee	47
Texas	48
Utah	49
Vermont	50
Virginia	51
Washington	53
West Virginia	54
Wisconsin	55
Wyoming	56
Gujarat	88
Kerela	999

Back To HOME!!!

6 QUERY EXECUTION

In this section, we include four queries. Additionally, we also include two trigger queries

- The Problem statement for the following query is given below, We find the University id and University name with the Highest number of students. This query is useful as it provides insight into choosing the most popular universities among students.

```
-- Query1==> To find the University id and University name with Highest number of students.
EXPLAIN ANALYSE select university_details.universityid,
university_details.university_name,university_award_details.student_count
from university_details
join university_award_details
ON university_award_details.universityid = university_details.universityid
where university_award_details.student_count=(
                                        select max(u1.student_count)
                                        from university_award_details u1);
```

- The Problem statement for the following query is given below, We find the total number of Males in California along with their stateid. This is a sample query to check the gender diversity at universities in California.

```
-- Query 2 ==> Find total number of Male in California along with their stateid.
EXPLAIN ANALYSE SELECT state_details.stateid,state_details.state,
      state_student_data.gender,COUNT(state_student_data.gender)
      AS Gender_count
      FROM state_details JOIN state_student_data USING(state)
      WHERE state_student_data.gender='Male'
      AND state_details.state='California'
      GROUP BY state_details.stateid,state_details.state,state_student_data.gender;
```

- The Problem statement for the following query is given below, We find the specific college type starting with research universities, and find the universityid,name of university along with passing rate in decreasing order.

```
-- --Query 3 ==> For specific college type starting with research universities,
--               find the universityid,name of university along with passing
--               rate in its decreasing order.
EXPLAIN ANALYSE SELECT university_details.universityid,university_details.university_name,
      university_demographic_details.specific_college_type,
      university_progress_details.passing_rate
      FROM university_details,university_demographic_details,
      university_progress_details
      WHERE university_details.universityid = university_demographic_details.universityid
      AND university_details.universityid = university_progress_details.universityid
      AND university_demographic_details.specific_college_type LIKE'Research%'
      ORDER BY university_progress_details.passing_rate DESC;
```

- The Problem statement for the following query is given below, Find states having more than 50 universities in descending order based on universities count

```
-- query 4 ==> Find states having more than 50 universities in their descending order based on
--              universities count
EXPLAIN ANALYSE SELECT university_award_details.state,
      count(university_details.universityid)
      AS Universities_Count
      FROM university_award_details
      INNER JOIN university_details
      ON university_details.universityid = university_award_details.universityid
      GROUP BY university_award_details.state
      HAVING count(university_details.universityid) >50
      ORDER BY count(university_details.universityid) DESC ;

```

- The Problem statement for the following query is given below, Creating a trigger 1.

```
-- Trigger 1 for insertion of records in student university details on insertion in state_student_data.
CREATE TRIGGER state_univ_insert AFTER insert ON state_student_data
for each row execute procedure Insert_records();

CREATE or REPLACE FUNCTION Insert_records() RETURNS TRIGGER as $state_univ_insert$
BEGIN
insert into state_university_details VALUES (NEW.state,NEW.organization_type,NEW.level);
RETURN NEW;
END;
$state_univ_insert$ LANGUAGE plpgsql;

SELECT * from state_university_details;
SELECT * from state_student_data;

insert INTO state_student_data values('Alaska','Presidency','2-year','20','32.1','78.0');
```

Now, we insert data to the state_student_data table where the trigger is created.

```
insert INTO state_student_data values('New York','University at Buffalo','2-year','20','32.1','78.0'
Data output  Messages  Notifications
INSERT 0 1
Query returned successfully in 73 msec.
```

Insertion successful on table state_Student_data where the Trigger is created.

```
insert INTO state_student_data values('New York','University at Buffalo','2-year','20','32.1','78.0'
SELECT * from state_university_details where organization_type='University at Buffalo';
SELECT * from state_student_data where organization_type='University at Buffalo';
Data output  Messages  Notifications
```

	state [PK] character varying (40)	organization_type [PK] character varying (30)	level [PK] character varying (20)	year [PK] integer	gender [PK] character varying (15)	race [PK] ch
1	New York	University at Buffalo	2-year	20	32.1	78.0

Insertion is done to the reference child table in Trigger Function. Data is displayed as above with three columns referenced from a parent but the other three columns left as NULL.

```
SELECT * from state_university_details where organization_type='University at Buffalo';
SELECT * from state_student_data where organization_type='University at Buffalo';
```

Data output | Messages | Notifications

	state character varying (40)	organization_type character varying (30)	level character varying (20)	university_count integer	counted_pct double precision	awards_per_state_value double precision
1	New York	University at Buffalo	2-year	[null]	[null]	[null]

- Creating a trigger 2.

```
-- Trigger 2 for insertion of records in university_demographic_bcnf1w on insertion in state_student_data.
CREATE TRIGGER univid_insert AFTER insert ON university_details
for each row execute procedure Insertrecords();

CREATE or REPLACE FUNCTION Insertrecords() RETURNS TRIGGER as $univid_insert$
BEGIN
insert into university_demographic_bcnf1w VALUES (NEW.universityid);
RETURN NEW;
END;
$univid_insert$ LANGUAGE plpgsql;

SELECT * from university_details order by universityid;
SELECT * from university_demographic_bcnf1w order by universityid;

-- insert INTO university_demographic_bcnf1w values('12345','Banglore');
INSERT into university_details values(12364,'Presidency university yelahanka');
delete from university_details where universityid = 12345;
```

insertion into University_details, where the trigger is created.

```
INSERT into university_details values(10234,'university at Buffalo');
delete from university_details where universityid = 123;
```

Data output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 66 msec.
```

Displaying the University_details table to verify that insetion is done for this table where we have created the triggers.

```
SELECT * from university_details order by universityid;
SELECT * from university_demographic_bcnf1w order by universityid;
```

Data output | Messages | Notifications

	universityid [PK] integer	university_name character varying (200)
1	10234	university at Buffalo
2	100654	Alabama A&M University
3	100663	University of Alabama ...
4	100690	Amridge University
5	100706	University of Alabama ...
6	100724	Alabama State Univers...

As we have referenced to university_demographic_bcnf1w in the trigger created Function, so now displayed the data to verify whether it's successfully inserted in the child table.

```
SELECT * from university_demographic_bcnf1w order by universityid;
```

Data output | Messages | Notifications

	universityid [PK] integer	website character varying (300)
1	10234	[null]
2	100654	www.aamu.edu/
3	100663	www.uab.edu
4	100690	www.amridgeuniversity...
5	100706	www.uah.edu
6	100724	www.alasu.edu/email/i...

```
END;
```

Similarly, We have trigger the Updation query and Trigger deletion Query.

- Trigger Update

```
-- Update trigger query --
CREATE TRIGGER update_state AFTER UPDATE ON state_details
for each row execute procedure Update_records();

CREATE or REPLACE FUNCTION Update_records() RETURNS TRIGGER as $update_state$
BEGIN
insert into university_demographic_bcnf4sc VALUES (new.state);
RETURN NEW;
END;
$update_state$ LANGUAGE plpgsql;

-- Update Query --
update state_details set state='Karnataka' where stateid=100;

-- Showing tables --
select * from state_details order by stateid DESC;
select * from university_demographic_bcnf4sc where state like'Kar%';
```

- Trigger Deletion

```
-- delete Trigger for state
CREATE TRIGGER delete_state AFTER DELETE ON state_details
for each row execute procedure Delete_records();

CREATE or REPLACE FUNCTION Delete_records() RETURNS TRIGGER as $delete_state$
BEGIN
DELETE from university_demographic_bcnf4sc where state=new.state;
RETURN NEW;
END;
$delete_state$ LANGUAGE plpgsql;

-- delete query
delete from state_details where state like 'Kar%';
```

7 QUERY OPTIMIZATION

For the queries mentioned in the previous section, we introduce indexing to optimize its performance.

For the first query, we create an index for university details and university_award_details.

```
-- Query1==>
EXPLAIN ANALYSE select university_details.universityid,
university_details.university_name,university_award_details.student_count
from university_details
join university_award_details
ON university_award_details.universityid = university_details.universityid
where university_award_details.student_count=(
                                        select max(u1.student_count)
                                        from university_award_details u1);

-- Query optimisation to improve performance on query 1
CREATE INDEX universityDetails on university_details(universityid,university_name);
CREATE INDEX universityAwardDetails on university_award_details(universityid,state);
```

We can see the performance increase with the index implementation.

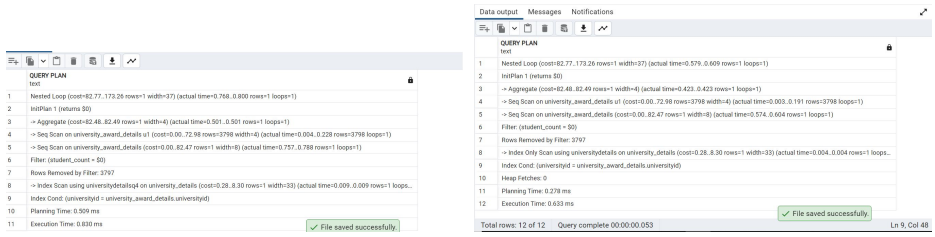


Fig. 1. Query Result 1

For the second query, we create an index for StateDetails and states_student_data.

```
-- Query 2 ==> Find total number of Male in California along with their stateid.
EXPLAIN ANALYSE SELECT state_details.stateid,state_details.state,
      state_student_data.gender,COUNT(state_student_data.gender)
      AS Gender_count
      FROM state_details JOIN state_student_data USING(state)
      WHERE state_student_data.gender='Male'
      AND state_details.state='California'
      GROUP BY state_details.stateid,state_details.state,state_student_data.gender;

-- Query optimisation to improve performance on query 2
CREATE INDEX StateDetails on state_details(stateid,state);
CREATE INDEX statestudentdata on state_student_data(gender);
```

We can see the performance increase with the index implementation.

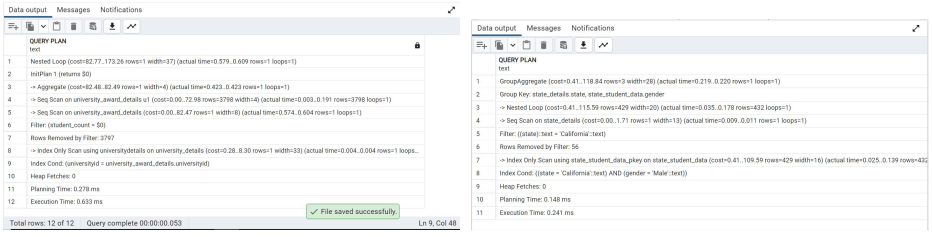


Fig. 2. Query Result 2

For the third query, we create an index for universityDemographicDetails and universityProgressDetails

```
-- --Query 3 ==> For specific college type starting with research universities,
--             find the universityid,name of university along with passing
--             rate in its decreasing order.
EXPLAIN ANALYSE SELECT university_details.universityid,university_details.university_name,
      university_demographic_details.specific_college_type,
      university_progress_details.passing_rate
      FROM university_details,university_demographic_details,
      university_progress_details
      WHERE university_details.universityid = university_demographic_details.universityid
      AND university_details.universityid = university_progress_details.universityid
      AND university_demographic_details.specific_college_type LIKE'Research%'
      ORDER BY university_progress_details.passing_rate DESC;

-- Query optimisation to improve performance on query 3
CREATE INDEX universityDemographicDetails on university_demographic_details(specific_college_type);
CREATE INDEX universityProgressDetails on university_progress_details(passing_rate);

drop index universityDemographicDetails;
drop index universityprogressdetails;
```

We can see the performance increase with the index implementation.

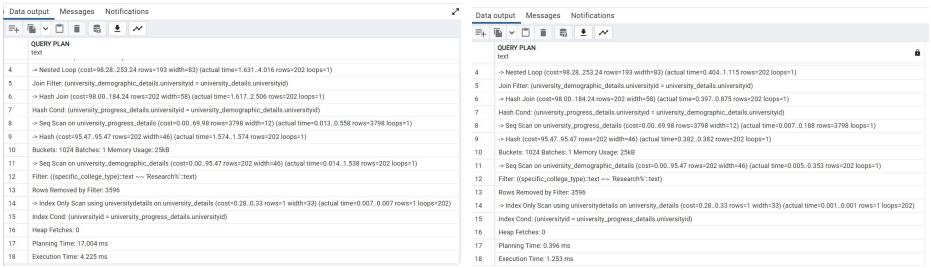


Fig. 3. Query Result 3

For the fourth query, we create an index for university details and university_award_details.

```
-- query 4 ==> Find states having more than 50 universities in their descending order based on
--             universities count.
EXPLAIN ANALYSE SELECT university_award_details.state,
      count(university_details.universityid)
      AS Universities_Count
      FROM university_award_details
      INNER JOIN university_details
      ON university_details.universityid = university_award_details.universityid
      GROUP BY university_award_details.state
      HAVING count(university_details.universityid) >50
      ORDER BY count(university_details.universityid) DESC ;

CREATE INDEX universityDetails on university_details(universityid,university_name);
CREATE INDEX universityAwardDetails on university_award_details(universityid,state);
```

We can see the performance increase with the index implementation.

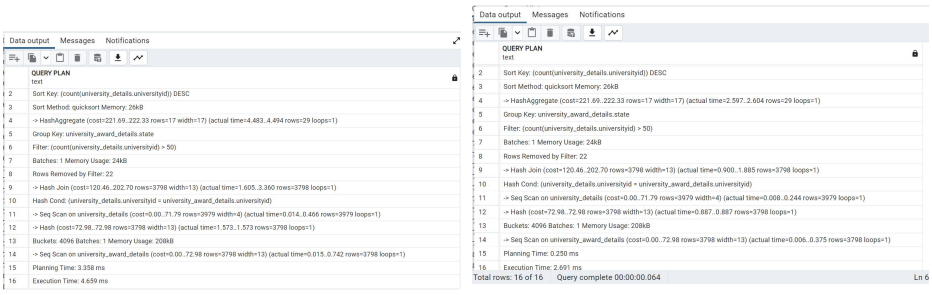


Fig. 4. Query Result 4

8 FUTURE SCOPE

The database for the student completion data-set is still in its nascent phases. This effort of making student degree completion rate public started by Bill and Melinda Gates foundations could be carried on by universities and governments

We can finally summarize the Query Optimization from the table below.

Table 1. Query performance

Query #	Performance Before in ms	Performance After in ms	Percentage Decrease in time
1	0.830	0.633	23.7349%
2	0.390	0.241	38.2051%
3	4.225	1.253	70.3432%
4	4.659	2.691	42.2408%

alike. Makings such data-sets publicly available makes not-only student make better choices but also promotes better recourse allocation by universities and state government. Additionally, this data-set being maintained by governing body, could promote better problems beings discussed and can give rise to understanding college dropout rates and address such issues in a more practical way.

9 REFERENCES

- Data.world. 2022. College Completion - dataset by databeats. [online] Available at: <https://data.world/databeats/college-completion> [Accessed 16 October 2022]